Διπλωματική Εργασία

Μελέτη του προβλήματος τοποθέτησης ελεγκτών σε δίκτυο καθορισμένο από λογισμικό, με χρήση του λογισμικού OpenDaylight

UNIVERSITY OF THESSALY

Συγγραφέας: Μάριος Καρατίσογλου
Επιβλέπων καθηγητής: Αθανάσιος Κοράκης

Εισαγωγή

Το Software-defined networking (SDN) είναι μια από τις μεγαλύτερες επαναστάσεις στον τομέα των τηλεπικοινωνιακών δικτύων. Αυτό γιατί άλλαξε ριζικά τον τρόπο με τον οποίο σχεδιάζονται τέτοια δίκτυα. Μερικά από τα πιο σημαντικά χαρακτηριστικά που επηρεάστηκαν είναι το control plane και το data plane. Το control plane (δηλαδή το πως διαχειρίζεται η κίνηση μέσα στο δίκτυο) και το data plane (δηλαδή το πως η κίνηση προωθείται ανάλογα με τις αποφάσεις του προηγούμενου) ήταν ενσωματωμένα στις συσκευές δικτύου (π.χ. router and switches), το οποίο είχε σαν αποτέλεσμα να μειώνει την ελαστικότητα ολόκληρης της δικτυακής υποδομής. Με μία αρχιτεκτονική καθορισμένη από λογισμικό, το control plane διαχωρίζεται πλέον από το data plane. Έτσι, ένα πρόγραμμα ελέγχου που λέγεται ελεγκτής (controller) ελέγχει την λειτουργία πολλών στοιχείων του data plane [1]. Επομένως, το SDN μπορεί να δώσει λύσεις σε πολλά προβλήματα που εμφανίζονται σε δίκτυα μεγάλης κλίμακας, όπως στην επεκτασιμότητα και διαχείριση του δικτύου.

Ο controller είναι το πιο σημαντικό μέρος ενός SDN δικτύου, εφόσον ολόκληρη η control plane αρχιτεκτονική είναι κεντρικοποιημένη σε αυτό. Ωστόσο, ένα δίκτυο με έναν μόνο controller επιφέρει νέες αδυναμίες, όπως έλλειψη αξιοπιστίας. Σε περίπτωση προβλήματος στην λειτουργία του controller, παρουσιάζεται κατάρρευση ολόκληρου του control plane. Προς αντιμετώπιση αυτού του προβλήματος, το OpenDaylight (ODL) [2], έχει υλοποιημένο έναν μηχανισμό συσσωμάτωσης (clustering) στον οποίο λειτουργούν πολλαπλοί controllers, πετυχαίνοντας υψηλή διαθεσιμότητα και ανεκτικότητα σε σφάλματα. Ένα τέτοιος μηχανισμός κρίνεται απαραίτητος για την ορθή λειτουργία ενός τέτοιου δικτύου.

Ωστόσο, θέτοντας σε λειτουργία πολλαπλούς controllers εισάγονται νέα προβλήματα, καθώς αυτοί θα πρέπει να λειτουργούν ως ένας ενιαίος controller με κεντρικοποιημένη λογική. Όταν πολλοί controllers λειτουργούν μαζί, η συνολική κατάσταση του συμπλέγματος είναι κατανεμημένη μεταξύ τους. Επίσης, η κατάσταση ενός ατομικού controller δεν θα πρέπει να διαφέρει από αυτή των υπολοίπων ελεγκτών του συμπλέγματος. Γι' αυτό το λόγο είναι απαραίτητος ένας μηχανισμός ο οποίος θα συγχρονίζει την λειτουργία των controllers του συμπλέγματος. Το ODL υλοποιεί διάφορες τεχνικές και αλγορίθμους για να απευθυνθεί σε αυτά τα προβλήματα, οι οποίες παρουσιάζονται στις επόμενες ενότητες. Το μειονέκτημα είναι πως η συσσωμάτωση πολλαπλών controllers καταναλώνει επιπλέον bandwidth, καθώς εισάγει επιπλέον κίνηση μεταξύ των ελεγκτών.

Η ελαχιστοποίηση του bandwidth που απαιτείται για την κίνηση που παράγεται από τους controllers είναι ο στόχος με την μεγαλύτερη προτεραιότητα, ειδικά σε δίκτυα με χαμηλής χωρητικότητας, όπως ασύρματα δίκτυα χαμηλής ταχύτητας με in-band έλεγχο [3]. Επιπλέον,

λιγότερη κίνηση από ελεγκτές σημαίνει κατανάλωση λιγότερης ενέργειας για non-data μεταδόσεις, που είναι πολύ σημαντικό για δίκτυα χαμηλής ενέργειας. Για όλους αυτούς τους λόγους, η ελαχιστοποίηση της επιπλέον κίνησης από τους ελεγκτές είναι μεγάλης σημασίας.

Ο σκοπός αυτής της έρευνας είναι να περιγράψει την επικοινωνία μεταξύ των ODL ελεγκτών που βρίσκονται στο ίδιο σύμπλεγμα. Ένα μοντέλο δικτύου εισάγεται για ελαχιστοποίηση της συνολικής κίνησης που παράγεται από τους ελεγκτές μέσα στο σύμπλεγμα. Τελικά, τα αποτελέσματα των πειραμάτων αναδεικνύουν πρότυπα της χρήσης bandwidth τόσο για την επικοινωνία μεταξύ των ελεγκτών, όσο και για την επικοινωνία μεταξύ ελεγκτών και switches.

Software Defined Networking

Ορισμός του SDN

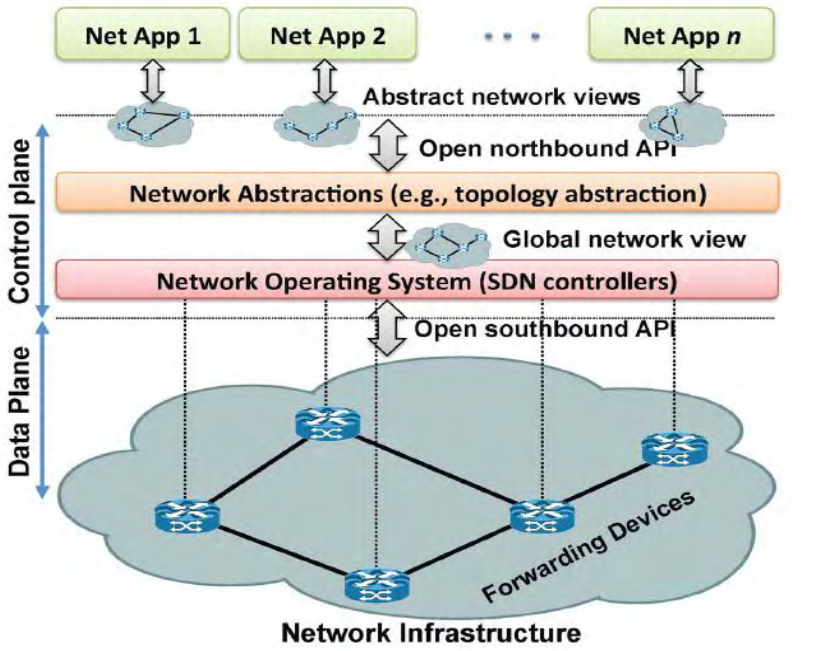
Ο όρος Software Defined Networking αρχικά ορίστηκε για να περιγράψει τις ιδέες και την δουλεία γύρω από το OpenFlow στο Stanford University [4]. Αναφέρεται σε μια αρχιτεκτονική δικτύου όπου η κατάσταση προώθησης στο data plane καθορίζεται από ένα απομακρυσμένα ελεγχόμενο plane. Τα 4 βασικά στοιχεία που ορίζουν μια αρχιτεκτονική ως SDN είναι:

1. Τα control και data planes είναι πλέον αποσυνδεδεμένα μεταξύ τους. Η λειτουργία ελέγχου έχει αφαιρεθεί από τις συσκευές δικτύου, οι οποίες πλέον είναι απλά στοιχεία προώθησης.
2. Οι αποφάσεις προώθησης βασίζονται στα flows. Ένα flow ορίζεται ως ένα set τιμών που λειτουργούν ως κριτήρια αντιστοίχισης περιεχομένου ενός πακέτου σε ένα συγκεκριμένο σύνολο εντολών. Έτσι πετυχαίνεται η ενοποίηση της συμπεριφοράς διαφορετικών τύπων συσκευών δικτύου, συμπεριλαμβανομένων routers, switches, firewalls κ.α.
3. Η λογική ελέγχου βρίσκεται σε μια εξωτερική οντότητα, τον SDN ελεγκτη. Ο ελεγκτής είναι μια πλατφόρμα λογισμικού η οποία παρέχει την δυνατότητα προγραμματισμού των συσκευών δικτύου.
4. Το δίκτυο προγραμματίζεται μέσω εφαρμογών που τρέχουν πάνω στον ελεγκτή, ο οποίος αλληλεπιδρά με τις data plane συσκευές που του αντιστοιχούν.

Βασικά Χαρακτηριστικά ενός SDN

Μιλώντας για δίκτυο καθορισμένο από λογισμικό, ουσιαστικά γίνεται αναφορά σε ένα σύνολο από χαρακτηριστικά που το καθορίζουν. Αυτά τα χαρακτηριστικά αποτελούν τον κορμό κάθε SDN, καθορίζονται ως αναπόσπαστο κομμάτι του και είναι τα εξής:

1. Συσκευές προώθησης (forwarding devices): Αυτές οι συσκευές μπορεί να είναι υλοποιημένες είτε σε hardware είτε σε software. Ακολουθούν ένα σύνολο από εντολές (flow rules) με τις οποίες κάνουν τις αντίστοιχες δράσεις πάνω στα εισερχόμενα πακέτα. Αυτές οι εντολές ορίζονται από southbound διεπαφές (π.χ. Openflow) και τα εγκαθιστούν οι ελεγκτές που υλοποιούν αυτές τις διεπαφές.
2. Data plane: Συσκευές προώθησης που συνδέονται είτε μέσω ασύρματων καναλιών επικοινωνίας είτε μέσω συνδέσμων καλωδίων. Η υποδομή δικτύου περιλαμβάνει τις διασυνδεδεμένες συσκευές προώθησης, οι οποίες αντιπροσωπεύουν το data plane.
3. Southbound Interface: Το σετ των εντολών μιας συσκευής προώθησης ορίζεται από το southbound API, το οποίο είναι μέρος του southbound interface. Επιπλέον, ορίζει το πρωτόκολλο επικοινωνίας μεταξύ των συσκευών προώθησης και των στοιχείων στο control plane. Αυτό το πρωτόκολλο καθορίζει τον τρόπο με τον οποίο τα στοιχεία του data και control plane αλληλεπιδρούν μεταξύ τους.
4. Control Plane: Συσκευές προώθησης προγραμματίζονται από το control plane μέσω του southbound interface. Αποτελεί τον “εγκέφαλο” του δικτύου. Όλη η λογική ελέγχου βρίσκεται στους ελεγκτές, οι οποίοι αποτελούν το control plane.
5. Northbound interface: το λειτουργικό σύστημα δικτύου (π.χ. το Openflow) παρέχει ένα API στους προγραμματιστές εφαρμογών. Προσαρμόζει τις εντολές χαμηλού επιπέδου που χρησιμοποιούνται στο southbound interface κάνοντας τον προγραμματισμό τους πιο φιλικό προς τον χρήστη.
6. Management Plane: είναι ένα σετ από εφαρμογές που χρησιμοποιούν λειτουργίες που προσφέρει το northbound interface για να υλοποιήσουν ελέγχους στο δίκτυο. Τέτοιες εφαρμογές περιέχουν routing, firewalls, load balancers, monitoring κ.α.



SDN Αρχιτεκτονική και τα βασικά της χαρακτηριστικά [4]

OpenFlow

Το OpenFlow είναι το βασικό πρωτόκολλο επικοινωνίας μεταξύ του data plane και του control plane. Επιτρέπει την διαχείριση των συσκευών προώθησης όπως routers και switches.

Αυτό το πρωτόκολλο τερματίζει το πρόβλημα των στατικών αρχιτεκτονικών δικτύου. Με το OpenFlow είναι δυνατό να δημιουργήσει κανείς μια πολιτική ελέγχου του δικτύου η οποία θα υιοθετηθεί από ολόκληρο το δίκτυο, επιτρέποντας σε έναν κεντρικό ελεγκτή να διαχειρίζεται απομακρυσμένα την πληροφορία που προωθείται μέσω του data plane. Αυτό είναι μία πολύ σημαντική προσέγγιση, καθώς κάνει το δίκτυο πιο αυτόματο, εξαλείφοντας το πρόβλημα της ρύθμισης κάθε συσκευής μία προς μία χειροκίνητα. Ένα ακόμα σημαντικό χαρακτηριστικό είναι ότι το πρωτόκολλο δεν είναι vendor-specific, κάνοντας την διαδικασία πιο εύκολη.

OpenDayLight

Το OpenDayLight (ODL) είναι ένα project ανοιχτού κώδικα, ιδρύθηκε τον Απρίλιο του 2013 και η πρώτη έκδοση έγινε διαθέσιμη τον Φεβρουάριο του 2014. Δημιουργήθηκε με βασικό σκοπό να μειώσει το “κλείδωμα” των συσκευών συγκεκριμένων πωλητών με την χρήση του OpenFlow και μετέπειτα να υποστηρίζει περισσότερα πρωτόκολλα από το OpenFlow.

Ο στόχος του OpenDaylight είναι να παρέχει ένα κεντρικοποιημένο σύστημα διαχείρισης που ελέγχει ένα προγραμματίσιμο δίκτυο. Αυτό επιτυγχάνεται χρησιμοποιώντας API frameworks. Αποτελεί επίσης μια ελαστική SDN πλατφόρμα για δίκτυα ανεξαρτήτου μεγέθους και κλίμακας, παρέχοντας υπηρεσίες στο δίκτυο ασχέτως του υλικού ή των πολλαπλών πωλητών. Η αρχιτεκτονική του επιτρέπει στους χρήστες να ελέγχουν τις εφαρμογές, τα πρωτόκολλα και να παρέχουν συνδέσεις σε εξωτερικούς πελάτες ή παρόχους [5].

Όλα τα συμβατικά δίκτυα πρέπει να τροποποιηθούν χειροκίνητα για να προσαρμοστούν στις ανάγκες του στιγμιαίου φόρτου εργασίας. Σε ένα SDN δίκτυο, ο OpenDayLight ελεγκτής αποτελεί μια πλατφόρμα η οποία αντιμετωπίζει τέτοιες προκλήσεις. Με την χρήση διάφορων API καταφέρνει να κάνει το δίκτυο πιο προγραμματίσιμο και προσαρμόσιμο σε στιγμιαίες αλλαγές. Στην διάθεσή του βρίσκονται πολλαπλές υπηρεσίες και πρωτόκολλα για την επίλυση πολλών διαφορετικών προβλημάτων.

Το γεγονός ότι ο κώδικας του ODL είναι ανοιχτός, αποτελεί το κλειδί για την ταχύτατη ανάπτυξη του. Πολλοί προγραμματιστές ανά τον κόσμο μπόρεσαν να συνεισφέρουν στην ανάπτυξη λογισμικού για αυτό το σύστημα [6].

Αρχιτεκτονική

Ένας OpenDayLight ελεγκτής διαχωρίζεται σε τρία στρώματα, το άνω, το μεσαίο και το κάτω στρώμα, όπως φαίνεται στην Εικόνα 2.

Άνω στρώμα - Northbound interface:

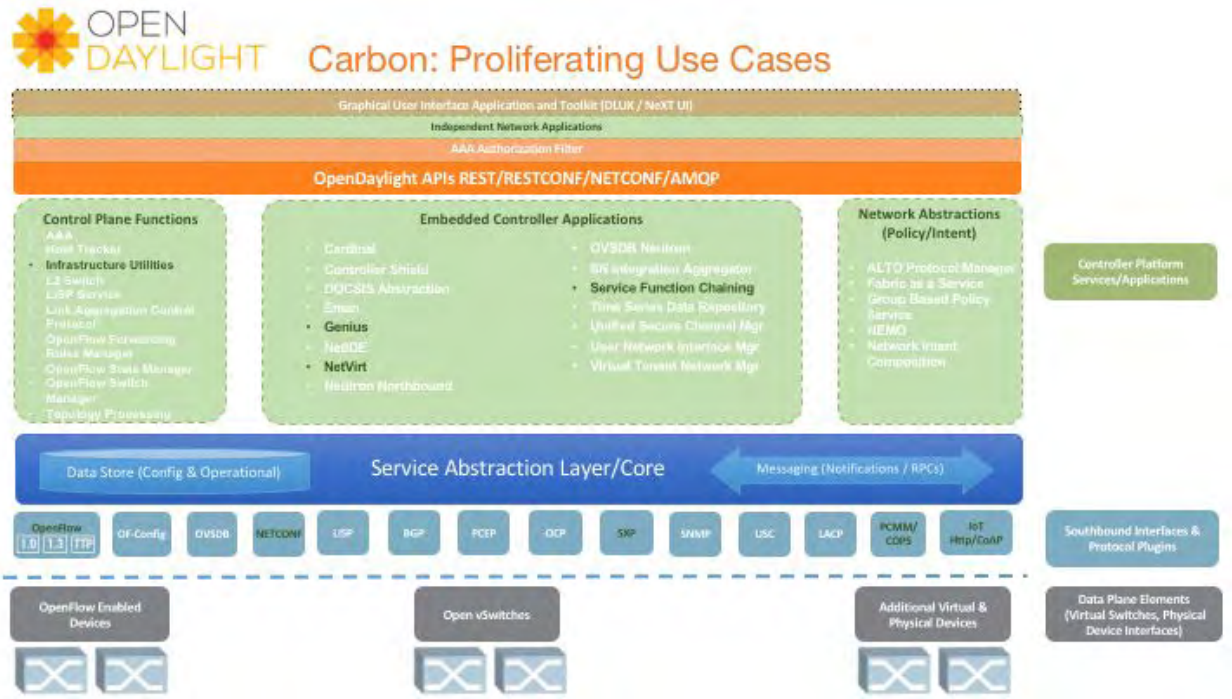
Σε αυτό το στρώμα, το northbound interface παρέχει υπηρεσίες του ελεγκτή και REST APIs. Αυτό βοηθάει στην διαχείριση των ρυθμίσεων ολόκληρης της δικτυακής υποδομής

Μεσαίο στρώμα - Πλατφόρμα ελεγκτή:

Σε αυτό το στρώμα, ο ελεγκτής επικοινωνεί με την δικτυακή υποδομή μέσω ορισμένων southbound plug-ins. Αυτό γίνεται ώστε να μπορεί να παρέχει δικτυακές υπηρεσίες, όπως διαχείριση της τοπολογίας και των switches.

Κάτω στρώμα - Southbound interface:

Σε αυτό το στρώμα βρίσκονται όλα τα πρωτόκολλα για διαχείριση και έλεγχο όλης της δικτυακής υποδομής. Περιέχει διάφορα plug-ins τα οποία υλοποιούν δικτυακά πρωτόκολλα, τα οποία επικοινωνούν απευθείας με το υλικό. Εδώ βρίσκεται το OpenFlow πρωτόκολλο.



Εικόνα 2 - Αρχιτεκτονική του ελεγκτή OpenDayLight [7]

Συσσωμάτωση (clustering) στο OpenDayLight

Με την χρήση του μηχανισμού συσσωμάτωσης που παρέχει το OpenDayLight, είναι δυνατόν πολλαπλοί ελεγκτές να λειτουργούν συγχρονισμένα, ως μία ενιαία οντότητα. Αυτή είναι μια από τις βασικότερες λειτουργίες του ODL και παρουσιάζει τα παρακάτω πλεονεκτήματα.

Το πρώτο πλεονέκτημα είναι ότι παρουσιάζει κλιμακώσιμη λειτουργία. Αν υπάρχουν περισσότεροι από ένας ελεγκτές στο δίκτυο, αυτό θα μπορεί να δουλέψει καλύτερα και να επεξεργαστεί περισσότερο όγκο δεδομένων. Τα δεδομένα μπορούν επίσης να κατανεμηθούν στους ελεγκτές.

Ένα δεύτερο πλεονέκτημα είναι η υψηλή διαθεσιμότητα του δικτύου. Αν ένας ελεγκτής αποσυνδεθεί από το δίκτυο λόγω σφάλματος, το δίκτυο θα μπορεί να συνεχίσει να λειτουργεί καθώς θα υπάρχουν επιπλέον διαθέσιμοι ελεγκτές.

Το τρίτο πλεονέκτημα είναι η συντήρηση των δεδομένων. Σε περίπτωση βλάβης ενός ελεγκτή τα δεδομένα που αυτός είχε δεν θα χαθούν, καθώς αυτά είναι κατανεμημένα σε όλους τους ελεγκτές του συμπλέγματος.

Κατανεμημένη αποθήκη δεδομένων

Η αποθήκευση δεδομένων στο ODL χωρίζεται σε τρία αποκόμματα (shards), την καταγραφή (inventory), την τοπολογία (topology) και την φρυγανιέρα (toaster). Το inventory shard περιέχει τους κανόνες των flows που έχουν εγκατασταθεί στο σύμπλεγμα των ελεγκτών. Το topology shard περιέχει πληροφορία για την τοπολογία του δικτύου. Το μέγεθός του αυξάνεται καθώς προσθέτουμε περισσότερα αντικείμενα στο δίκτυο. Το toaster shard καθορίζει τις Διαδικασίες Απομακρυσμένης Κλήσης (RPC) που μπορούν να εκτελεστούν στο συγκεκριμένο shard, μαζί με τις πράξεις που πρέπει να γίνουν σε κάθε κλήση.

Κάθε shard είναι κατανεμημένο στο σύμπλεγμα των ελεγκτών. Αρχικά, ανατίθενται σε ένα συγκεκριμένο κόμβο και έπειτα, για να αποφευχθεί η απώλεια δεδομένων σε περίπτωση σφάλματος ενός κόμβου, κάθε shard αντιγράφεται σε όλους τους ελεγκτές του συμπλέγματος (replica). Επομένως, όλα τα shards και τα replicas πρέπει να είναι συγχρονισμένα μεταξύ τους, για να επιτευχθεί συνέπεια (consistency) στη λειτουργία του συμπλέγματος. Για την επίτευξη της συνέπειας, το ODL χρησιμοποιεί τον Raft αλγόριθμο ομοφωνίας [8].

Σύμπλεγμα Akka και το πρωτόκολλο Gossip

Το Akka cluster [9] παρέχει μια ανθεκτική σε σφάλματα και αποκεντρωμένη υπηρεσία συσσωμάτωσης. Αυτό το πετυχαίνει χρησιμοποιώντας το το πρωτόκολλο Gossip και έναν αυτόματο εντοπιστή σφαλμάτων. Κάθε στιγμή που ένας κόμβος προσπαθεί να συνδεθεί σε ένα Akka σύμπλεγμα, αρχίζει ένα 4-way-handshake, όπως το έχει ορίσει το Gossip πρωτόκολλο. Επικοινωνεί με τους κόμβους-seeders, οι οποίοι έχουν οριστεί από πριν μέσω του αρχείου ρυθμίσεων του Akka στέλνοντας ***InitJoin*** μηνύματα. Ο seeder επιβεβαιώνει το *InitJoin* αίτημα απαντώντας με ένα ***InitJoinAck*** μήνυμα. Ο κόμβος τώρα στέλνει ένα ***Join*** μήνυμα και συνδέεται επιτυχώς με το σύμπλεγμα αφού λάβει ένα ***Welcome*** μήνυμα ως απάντηση από τον seeder.

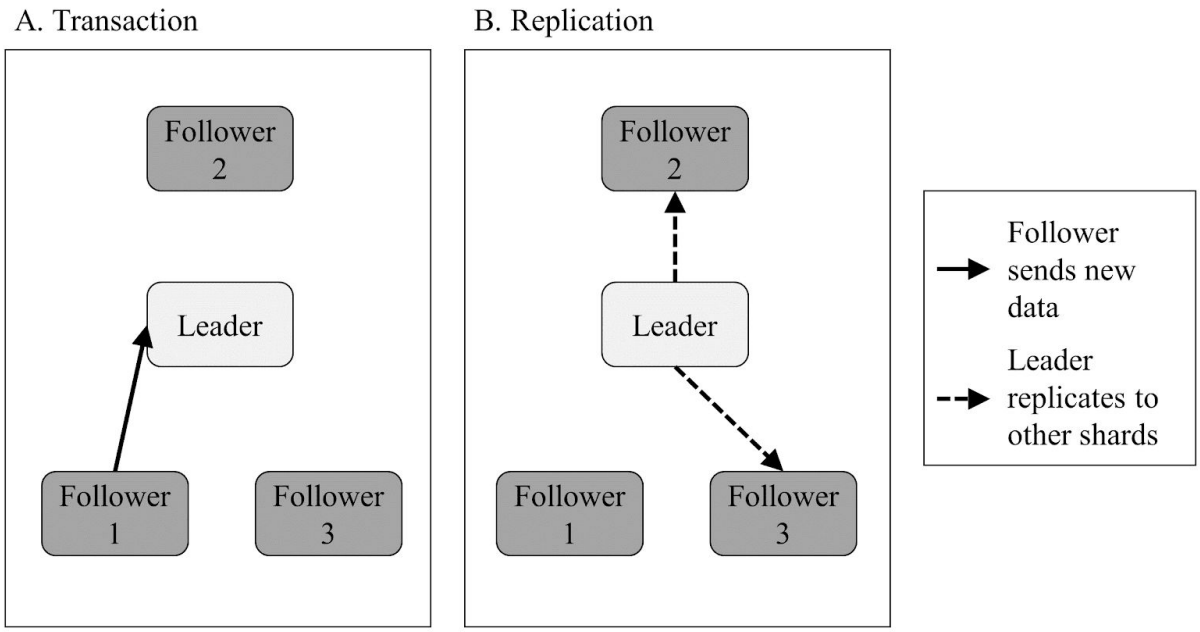
Το πρωτόκολλο Gossip είναι υπεύθυνο και για την επιμονή (persistency) του συμπλέγματος. ***Heartbeat*** μηνύματα ανταλλάσσονται περιοδικά μεταξύ των κόμβων του συμπλέγματος και αναμένεται μήνυμα ***HeartbeatRsp*** εντός ενός χρονικού ορίου για να θεωρηθεί ένας κόμβος ως ενεργός.

Raft αλγόριθμος ομοφωνίας

Ένας αλγόριθμος ομοφωνίας επιτρέπει σε ένα σύνολο κόμβων ή εξυπηρετητών να δουλεύουν σύγχρονα, σαν ένα μοναδικό σύστημα που είναι ικανό να χειρίζεται βλάβες σε επιμέρους κόμβους που το αποτελούν. Αυτό γίνεται αντιγράφοντας την κατάσταση του κόμβου-αρχηγού (leader). Το ODL χρησιμοποιεί τον Raft αλγόριθμο [7] για να πετύχει συνέπεια συστήματος, συγχρονίζοντας την κατάσταση των shards που βρίσκονται κατανεμημένα στο σύμπλεγμα ελεγκτών.

Σε κάθε shard, ο Raft αλγόριθμος αναθέτει μια από τις τρεις καταστάσεις, αρχηγός (leader), ακόλουθος (follower) και υποψήφιος (candidate). Ο Raft ξεκινάει κάνοντας μία ψηφοφορία ανάμεσα στους κόμβους του συμπλέγματος. Το shard το οποίο παίρνει τα πιο πολλά ***RequestVote*** μηνύματα γίνεται ο shard leader. Ο leader είναι υπεύθυνος να ανακαλύπτει ασυνέπειες μεταξύ των υπολοίπων shards και να αντιγράφει τα κατάλληλα δεδομένα στέλνοντας ***AppendEntries*** πακέτα. Επιπλέον, κενά *AppentEntries* μηνύματα στέλνονται περιοδικά από το Raft και χρησιμοποιούνται ως heartbeat μηνύματα. Ο leader πρέπει να απαντήσει με αντίστοιχο ***AppendEntriesRsp*** μήνυμα μέσα σε ένα ορισμένο χρονικό περιθώριο, αλλιώς θεωρείται ως ανενεργός και μια νεα ψηφοφορία ξεκινάει.

Στην περίπτωση όπου ένας follower έχει νέα πληροφορία, π.χ. ένα switch συνδέθηκε σε αυτόν, στέλνει ένα ***CreateTransaction*** μήνυμα στον leader, ειδοποιώντας τον για τα νέα δεδομένα που εισήλθαν στο σύμπλεγμα. Αφού ολοκληρωθεί αυτή η συναλλαγή, ο leader προωθεί αυτά τα δεδομένα στα υπόλοιπα αντίγραφα των shards, κρατώντας τα ενημερωμένα. Ο ρόλος του leader είναι πολύ βασικός, καθώς όλα τα shards προωθούν δεδομένα σε αυτόν και αυτός με την σειρά του είναι υπεύθυνος να τα προωθήσει στους υπόλοιπους followers. Απεικόνιση μίας τέτοιας συναλλαγής γίνεται στην Εικόνα 3. Όπως θα δούμε στα αποτελέσματα των πειραμάτων, η συνολική επικοινωνία του συμπλέγματος είναι κεντρικοποιημένη προς τον κόμβο-leader. Αυτό συμβαίνει γιατί όλοι οι shard leaders συγκεντρώνονται σε έναν κόμβο, συνήθως σε αυτόν που εισέρχεται πρώτος στο σύμπλεγμα, όπως έδειξε και η έρευνα στο [10]. Αυτό έχει ως αποτέλεσμα έναν κόμβο που έχει συγκεντρωμένους όλους τους shard leaders και τους υπόλοιπους να έχουν όλους τους shard followers.



Εικόνα 3 - Στο A, ο ακόλουθος 1 στέλνει στον Leader τα νέα δεδομένα του. Στη συνέχεια, στο B, ο Leader αντιγράφει τα νέα δεδομένα στους υπόλοιπους ακόλουθους. Παρατηρήστε ότι δεν υπάρχει άμεση αναπαραγωγή μεταξύ των ακόλουθων.

Μοντέλο Δικτύου

Αυτό που ακολουθεί σε αυτή την ενότητα είναι η περιγραφή της διαδικασίας δημιουργίας ενός μοντέλου δικτύου που αναπαριστά τη συνολική κίνηση ελέγχου (control traffic). Θεωρείται ένα μη κατευθυνόμενο γράφημα $\mathcal{G}=(\mathcal{S},\mathcal{L})$, το οποίο αντιπροσωπεύει το control plane ενός SDN δικτύου, όπου το $\mathcal{S}$ αντιπροσωπεύει το σύνολο των switches και το $\mathcal{L}$ αντιπροσωπεύει τις συνδέσεις μεταξύ τους. Θεωρούμε $S=|\mathcal{S}|$ και $L=|\mathcal{L}|$ το πλήθος των switches και το πλήθος των συνδέσμων αντίστοιχα. Χωρίς την απώλεια γενικότητας, υποθέτουμε ότι ο αλγόριθμος δρομολόγησης είναι ο αλγόριθμος συντομότερου μονοπατιού (shortest path), το μονοπάτι που συνδέει το ζεύγος των switches $s_1, s_2 \in \mathcal{S}$ είναι $p(s_1, s_2)$ και ο αριθμός των συνδέσμων που υπάρχουν σε αυτό το μονοπάτι είναι $|p(s_1, s_2)|$. Τέλος, $\mathcal{C} \subseteq \mathcal{S}$ είναι το υποσύνολο των switches όπου $C = |\mathcal{C}|$ ελεγκτές είναι τοποθετημένοι και οργανωμένοι σε ένα σύμπλεγμα, ενώ $c_l \in \mathcal{C}$ συμβολίζει τον αρχηγό (leader) του συμπλέγματος. Στο εξής, θα αναφερόμαστε στο $c \in \mathcal{C}$ ως έναν ελεγκτή, ή ως το αντίστοιχο switch που το φιλοξενεί. Το $c_s \in \mathcal{C}$ συμβολίζει τον ελεγκτή

που αντιστοιχεί το switch *s*. Το διάνυσμα **c** = ($c_l \in$ **C,** ($c_s \in$ **C** : s $\in$ ***S***)) περιγράφει μία τοποθέτηση ελεγκτών και ανάθεση των switches. Η πρώτη τιμή του διανύσματος δείχνει τον αρχηγό του συμπλέγματος. Κάθε επόμενη συντεταγμένη του διανύσματος αντιστοιχεί σε ένα switch s $\in$ ***S*** και η τιμή δείχνει τον αντίστοιχο $c_s \in$ **C** ελεγκτή του switch.

Η χρήση bandwidth για την κίνηση Ελεγκτή-προς-Switch (Ctr-Sw) για όλους τους συνδέσμους του δικτύου αναγράφεται ως,

$$B^S \triangleq \sum_{s \in \mathcal{S}} \sum_{h \in p(s, c_s)} b^s = \sum_{s \in \mathcal{S}} w^s b^s, \qquad (1)$$

όπου b^s είναι το Bandwidth που απαιτείται για την Ctr-Sw κίνηση που ανταλλάσσεται μεταξύ του switch s και του ελεγκτή c_s, ενώ $w^s = |p(s, c_s)|$ είναι το μήκος του μονοπατιού που συνδέει το switch s με τον ελεγκτή c_s. Παρόμοια, το Ctr-Ctr Bandwidth που χρησιμοποιείται σε όλους τους συνδέσμους του δικτύου είναι,

$$B^C \triangleq \sum_{c \in \mathcal{C} - \{c_l\}} \sum_{h \in p(c, c_l)} b^c = \sum_{c \in \mathcal{C} - \{c_l\}} w^c \left(b^c_{\text{inv}} + b^c_{\text{topo}} \right), \qquad (2)$$

όπου b^c είναι το Bandwidth που απαιτείται για την Ctr-Ctr κίνηση που ανταλλάσσεται μεταξύ του switch l και του ελεγκτή c_l, το οποίο είναι το άθροισμα της b^c_{inv} κίνησης που παράγεται από το inventory shard και της της b^c_{topo} κίνησης που παράγεται από το topology shard, ενώ $w^c = |p(c, c_l)|$ είναι το μήκος του μονοπατιού που ενώνει τους ελεγκτές c και c_l.

Αυτή η μελέτη έχει ως σκοπό να σχηματίσει ένα μαθηματικό πρόβλημα, το οποίο θα δώσει λύση στην βέλτιστη τοποθέτηση ελεγκτών **C***, την βέλτιστη τοποθέτηση του αρχηγού c^*_l και την βέλτιστη ανάθεση των switches, δεδομένου του διανύσματος **c*** = (c^*_l **,** ($c^*_s \in$ **C*** : s $\in$ ***S***)), το οποίο ελαχιστοποιεί το συνολικό bandwidth που απαιτείται για την κίνηση ελέγχου. Το πρόβλημα εκφράζεται ως,

$$\mathbf{c}^* \triangleq \arg\min_{\mathbf{c}} \left(\sum_{s \in \mathcal{S}} w^s b^s + \sum_{c \in \mathcal{C} - \{c_l\}} w^c \left(b^c_{\text{inv}} + b^c_{\text{topo}} \right) \right). \qquad (3)$$

Σε αυτό το σημείο, γίνονται οι ακόλουθες παρατηρήσεις που επιβεβαιώνονται από τα πειράματα με τους ODL ελεγκτές, οι οποίοι οργανώνονται σε ένα σύμπλεγμα και χρησιμοποιούν το OpenFlow ως το southbound interface τους και τα χαρακτηριστικά του L2 switch plug in προγράμματος για την ρύθμιση των flows. Περισσότερα για αυτά τα πειράματα ακολουθούν σε επόμενη ενότητα.

Παρατήρηση 1: *Το απαιτούμενο bandwidth για τη Ctr-Sw κίνηση που ανταλλάσσεται μεταξύ ενός switch και του ελεγκτή του είναι ανάλογο του αριθμού των flows που υπάρχουν σε αυτό το switch.*

Ορίζουμε το β^s να συμβολίζει το απαιτούμενο bandwidth για κάθε flow. Αν f^s είναι ο αριθμός των flows που υπάρχουν στο s $\in$ **S**, τότε $b^s = f^s \beta^s, \ \forall$ s $\in$ **S.**

Παρατήρηση 2: *Το απαιτούμενο bandwidth για τη Ctr-Ctr κίνηση που ανταλλάσσεται μεταξύ ενός ακόλουθου ελεγκτή και του αρχηγού, λόγω του inventory shard, είναι ανάλογο του συνολικού αριθμού των flows των switches που έχουν ανατεθεί στον ακόλουθο ελεγκτή. Είναι επίσης ανάλογο με τον συνολικό αριθμό των flows που υπάρχουν στο switches που έχουν ανατεθεί σε όλους τους υπόλοιπους ελεγκτές (περιλαμβάνοντας και τον αρχηγό).*

Ορίζουμε το β_{inv}^{int} να συμβολίζει το απαιτούμενο bandwidth για την σύνδεση μεταξύ ενός ελεγκτή και του αρχηγού, για κάθε flow που ρυθμίστηκε από αυτόν τον ελεγκτή και β_{inv}^{ext} να συμβολίζει το αντίστοιχο bandwidth για κάθε flow που ρυθμίστηκε από όλους τους υπόλοιπους ελεγκτές (συμπεριλαμβανομένου και το αρχηγού). Αν $f^c \triangleq \sum_{s \in \mathcal{S}: c_s = c} f^s$ είναι ο αριθμός όλων των flows που υπάρχουν στα switches που έχουν ανατεθεί στον ελεγκτή c $\in$ **C** και $F \triangleq \sum_{s \in \mathcal{S}} f^s$ είναι το συνολικό πλήθος όλων των flows, τότε $b_{inv}^c = f^c \beta_{inv}^{int} + (F - f^c)\beta_{inv}^{ext}, \ \forall$ c $\in$ **C**

Παρατήρηση 3: *Το απαιτούμενο bandwidth για τη Ctr-Ctr κίνηση που ανταλλάσσεται μεταξύ ενός ακόλουθου ελεγκτή και του αρχηγού, λόγω του topology shard, είναι ανάλογο του συνολικού αριθμού των switches που έχουν ανατεθεί στον ακόλουθο ελεγκτή. Είναι επίσης ανάλογο με τον συνολικό αριθμό των switches που έχουν ανατεθεί σε όλους τους υπόλοιπους ελεγκτές (περιλαμβάνοντας και τον αρχηγό).*

Ορίζουμε το β_{topt}^{int} να συμβολίζει το απαιτούμενο bandwidth για την σύνδεση μεταξύ ενός ελεγκτή και του αρχηγού, για κάθε flow που ρυθμίστηκε από αυτόν τον ελεγκτή και β_{topo}^{ext} να συμβολίζει το αντίστοιχο bandwidth για κάθε flow που ρυθμίστηκε από όλους τους υπόλοιπους ελεγκτές (συμπεριλαμβανομένου και το αρχηγού). Αν $y^c \triangleq \sum_{s \in \mathcal{S}: c_s = c} 1$ είναι ο αριθμός των switches που έχουν ανατεθεί στον ελεγκτή c $\in$ **C** τότε $b_{topo}^c = y^c \beta_{topo}^{int} + (S - y^c)\beta_{topo}^{ext}, \ \forall$ c $\in$ **C**

Βασιζόμενοι στις παραπάνω παρατηρήσεις, το πρόβλημα της εξίσωσης 3 τροποποιείται στο εξής,

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \Big(\sum_{s \in \mathcal{S}} w^s f^s \beta^s + \sum_{c \in \mathcal{C}} w^c \big(f^c \beta^{\text{int}}_{\text{inv}} + (F - f^c) \beta^{\text{ext}}_{\text{inv}} \big) + \sum_{c \in \mathcal{C}} w^c \big(y^c \beta^{\text{int}}_{\text{topo}} + (S - y^c) \beta^{\text{ext}}_{\text{topo}} \big) \Big) \Rightarrow \quad (4)$$

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \Big(\sum_{s \in \mathcal{S}} w^s f^s \beta^s + \sum_{c \in \mathcal{C}} w^c \big(f^c (\beta^{\text{int}}_{\text{inv}} - \beta^{\text{ext}}_{\text{inv}}) + y^c (\beta^{\text{int}}_{\text{topo}} - \beta^{\text{ext}}_{\text{topo}}) \big) \Big). \quad (5)$$

Το πρόβλημα 5 είναι ισοδύναμο με το Πρόβλημα 4 , καθώς οι ποσότητες $F\beta^{ext}_{topo}$ και $F\beta^{ext}_{inv}$ είναι ανεξάρτητες σταθερές, δεδομένου ενός δικτύου. Αυτό το πρόβλημα λύθηκε με Integer Quadratic Programming (IQP) και τα αποτελέσματα παρουσιάζονται σε επόμενη ενότητα.

Ελαχιστοποίηση της κίνησης ελέγχου

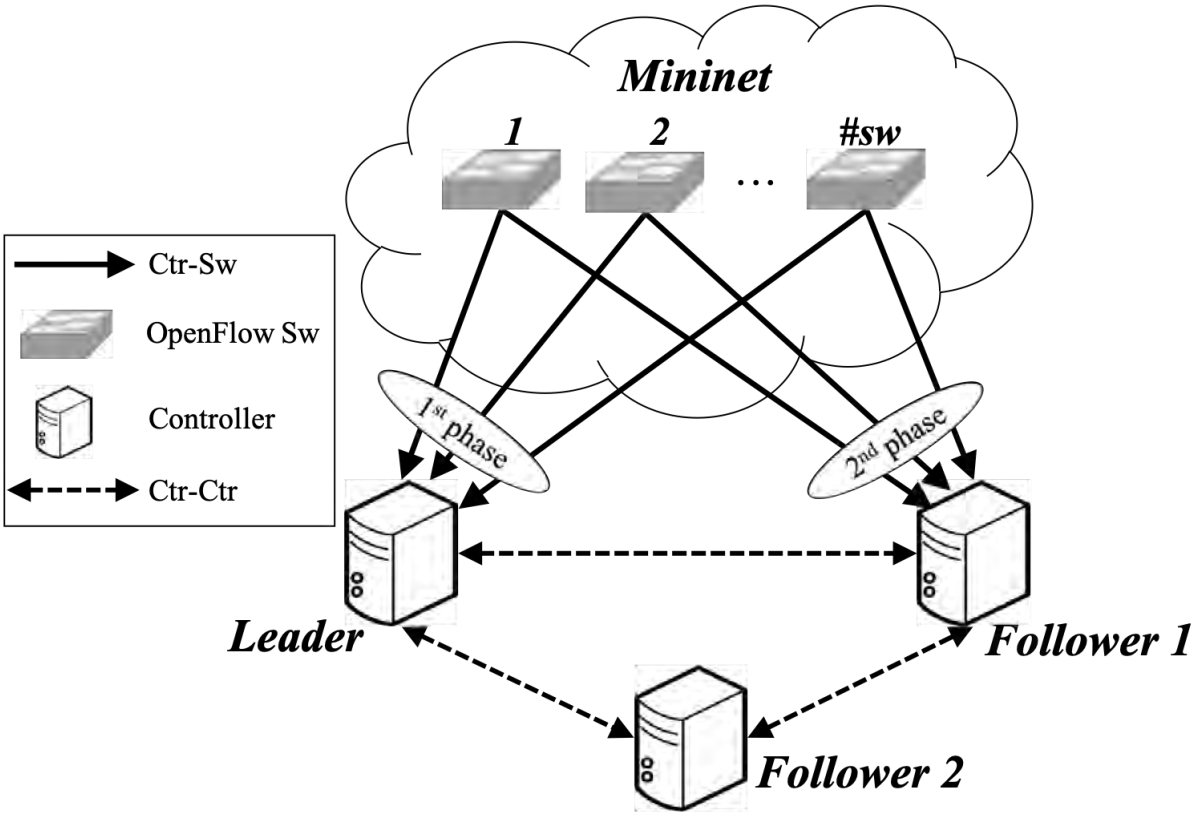
Ρύθμιση περιβάλλοντος πειραμάτων

Με σκοπό να μελετηθεί η επίδοση της επικοινωνίας μεταξύ των ελεγκτών του συμπλέγματος, χρησημοποιήθηκε το εργαστήριο πειραματισμών NITOS [11]. Ένα σύμπλεγμα ODL ελεγκτών χρησημοποιήθηκε ως control plane, ενώ ο προσομοιωτής δικτύου Mininet [12] χρησιμοποιήθηκε ως data plane. Κάθε ελεγκτής του συμπλέγματος έτρεξε σε ξεχωριστό κόμβο

στο NITOS. Ένα σύνολο τριών ελεγκτών χρησιμοποιήθηκε για να στηθεί το ODL σύμπλεγμα, όπου ο καθένας χρησιμοποιούσε το OpenFlow πρωτόκολλο και χαρακτηριστικά των λειτουργιών L2 Switch για την ρύθμιση των flows. Τελικά, χρησιμοποιήθηκε το iftop [13], ένα εργαλείο γραμμής εντολών, ανοιχτού λογισμικού, το οποίο παράγει μια συχνά ανανεωμένη λίστα των διαδικτυακών συνδέσεων, για την παρακολούθηση της επικοινωνίας μεταξύ των κόμβων του συμπλέγματος.

Πειραματισμοί πάνω στο Topology Shard

Όπως περιγράφηκε σε προηγούμενη ενότητα, το topology shard περιέχει πληροφορία σχετική με την τοπολογία του δικτύου. Όταν ένα topology shard σε ένα μέλος του συμπλέγματος ανανεώνει τα δεδομένα του, τα νέα δεδομένα αντιγράφονται στα υπόλοιπα μέλη. Για αυτό το σκοπό, μία Mininet τοπολογία ρυθμίστηκε με διάφορα πλήθη switches να συνδέονται σε κάθε ελεγκτή και μετρήθηκαν οι αλλαγές στο bandwidth που απαιτείται για τον συγχρονισμό όλων των topology shards. Οι τοπολογίες που ρυθμίστηκαν κατα την διάρκεια αυτών των πειραμάτων απεικονίζονται στην Εικόνα 4.



Εικόνα 4 - #sw switches είναι συνδεδεμένα είτε στον leader (1st phase) είτε στον Follower 1 (2nd phase), για να μετρηθεί η Ctr-Ctr κίνηση που δημιουργείται λόγω του topology shard.

#sw		*bandwidth usage (Mbps)*							
l	f_1	$l \to f_1$	$f_1 \to l$	$l \leftrightarrow f_1$	avg_1	$l \to f_2$	$f_2 \to l$	$l \leftrightarrow f_2$	avg_2
0	0	0.09	0.09	0.18	–	0.09	0.09	0.19	–
1	0	0.34	0.11	0.44	0.26	0.34	0.10	0.44	0.25
2	0	0.53	0.11	0.64	0.23	0.53	0.11	0.64	0.23
3	0	0.73	0.11	0.84	0.22	0.72	0.11	0.83	0.21
5	0	1.04	0.12	1.16	0.20	1.05	0.11	1.16	0.20
10	0	1.65	0.12	1.77	0.16	1.65	0.12	1.77	0.16
20	0	3.21	0.14	3.35	0.16	3.21	0.14	3.35	0.16
50	0	7.89	0.18	8.07	0.16	7.91	0.19	8.10	0.16
100	0	15.70	0.32	16.02	0.16	15.70	0.34	16.04	0.16
0	1	0.51	1.09	1.60	1.42	0.33	0.10	0.43	0.25
0	2	0.84	2.09	2.93	1.37	0.45	0.11	0.56	0.19
0	3	1.15	3.08	4.23	1.35	0.66	0.12	0.78	0.20
0	5	1.83	5.08	6.91	1.35	0.97	0.14	1.10	0.18
0	10	3.45	10.05	13.50	1.33	1.81	0.17	1.98	0.17
0	20	6.24	19.60	25.84	1.28	3.46	0.22	3.68	0.16
0	50	14.40	48.50	62.90	1.25	7.78	0.41	8.19	0.16
0	100	28.50	97.12	125.62	1.25	15.70	0.73	16.43	0.16

Πίνακας 1: Αποτελέσματα πειραμάτων στο topology shard.
Χρήση bandwidth για διάφορα πλήθη switches σε κάθε ελεγκτή

Ο Πίνακας 1 συνοψίζει τα αποτελέσματα αυτών των πειραμάτων. Η αριστερότερη στήλη αντιπροσωπεύει των αριθμό των switches (#sw) που είναι συνδεδεμένα σε κάθε ελεγκτή, όπου l αντιπροσωπεύει τον Leader και f1 τον Follower 1. Ο Follower 2 (f2) δεν έχει ποτέ συνδεδεμένο κάποιο switch σε αυτό. Οι στήλες με επιγραφή $x \to y$ παρουσιάζουν την χρήση bandwidth σε Mbps της κίνησης που στάλθηκε από το x στο y, όπου το x είναι είτε l, f1, f2, ενώ οι στήλες $l \leftrightarrow y$ δίνουν το συνολικό bandwidth usage για $l \to y$ και $y \to l$. Τελικά, οι στήλες avg1 και avg2 παρουσιάζουν την μέση αύξηση του Bandwidth για κάθε switch που προστίθεται και για τις δυο κατευθύνσεις, για τις επικοινωνίες μεταξύ $l \leftrightarrow f1$ και $l \leftrightarrow f2$ αντίστοιχα. Για παράδειγμα, όταν 2 και 3 switches είναι συνδεδεμένα στον leader, οι τιμές στην στήλη avg1 είναι $0,23 = (0,64 - 0,18)/2$ και $0,22 = (0,84 - 0,18)/3$ αντίστοιχα.

Στην πρώτη φάση των πειραμάτων, για να μελετηθεί το πως τα δεδομένα αντιγράφονται από το Leader shard στα follower shards, αυξήθηκε ο αριθμός των switches που είναι συνδεδεμένα στον leader. Τα αποτελέσματα αυτών των πειραμάτων παρουσιάζονται στο πάνω μισό του Πίνακα 1, με όλες τις γραμμές να έχουν μη μηδενικές τιμές κάτω από το l. Σημειώθηκε

αύξηση στην χρήση bandwidth για την επικοινωνία από την πλευρά του leader στους follower ελεγκτές. Δεν παρατηρήθηκε αύξηση στο bandwidth από την πλευρά των followers προς τον leader. Αυτό συμβαίνει διότι οι follower κόμβοι δεν έχουν καινούρια δεδομένα που χρειάζεται να αντιγραφούν. Η μέση αύξηση στο bandwidth ανά switch που παρατηρήθηκε συγκλίνει στα 0,16Mbps, είτε για $l \leftrightarrow f1$ είτε $l \leftrightarrow f1$ επικοινωνία.

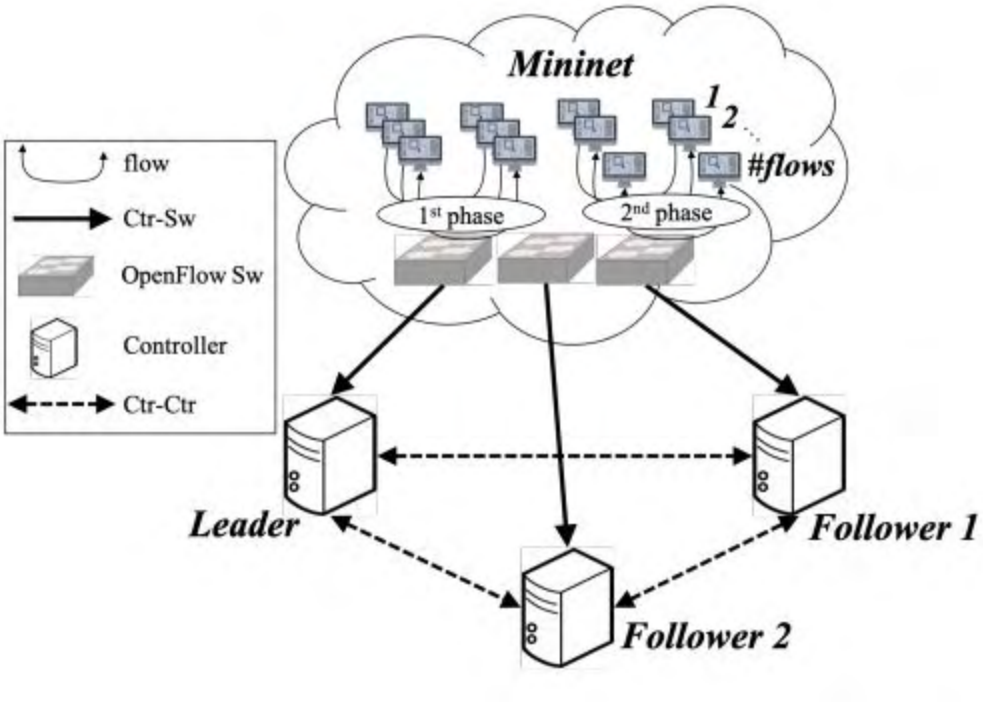
Στη δεύτερη φάση, αυξάνεται ο αριθμός των switches που είναι συνδεδεμένα στον Follower 1. Τα σχετικά αποτελέσματα βρίσκονται στο κάτω μισό του Πίνακα 1, με όλες τις γραμμές να έχουν μη μηδενικές τιμές κάτω από το f1. Σε αυτή τη περίπτωση, ο Follower 1 επικοινωνεί με τον Leader για να τον πληροφορήσει σχετικά με τα νέα δεδομένα τοπολογίας που εμφανίστηκαν στο δίκτυο και έπειτα ο Leader προβαίνει στην αντιγραφή των νέων δεδομένων προς τα παλαιότερα topology shards. Πράγματι, όπως μπορούμε να δούμε και στο Πίνακα 1, υπάρχει αύξηση στο bandwidth για την επικοινωνία από τον Follower 1 προς τον leader. Επιπλέον, αύξηση στο bandwidth υπάρχει στην επικοινωνία από τον Leader προς τον Follower 1 και Follower 2. Σε αυτή τη περίπτωση, η μέση αύξηση bandwidth για κάθε ένα switch που προστίθεται, είναι 1,25Mbps και 0,16Mbps για την $l \leftrightarrow f1$ και $l \leftrightarrow f2$ επικοινωνία αντίστοιχα.

Με βάση αυτά τα αποτελέσματα, επιβεβαιώνεται η Παρατήρηση 3, με $\beta_{topo}^{int} \approx 1,25Mbps$ και $\beta_{topo}^{ext} \approx 0.16Mbps$. Δεν υπάρχουν στήλες που να απεικονίζουν την επικοινωνία μεταξύ των 2 followers, καθώς είναι συνεχώς σταθερή και αμελητέα από άποψη κατανάλωσης bandwidth (περίπου 0,055Mbps), σε σύγκριση με τις υπόλοιπες επικοινωνίες. Αυτό οφείλεται στην αντιγραφή των shards και τον τρόπο που αυτή επιτυγχάνεται. Όταν ένα follower shard έχει νέα δεδομένα επικοινωνεί με το leader shard, και πλέον ο δεύτερος είναι υπεύθυνος για την προώθηση τους στους υπόλοιπους followers. Αυτό σημαίνει ότι δεν υπάρχει άμεση επικοινωνία μεταξύ των followers, σχετικά με την αντιγραφή των δεδομένων. Η μόνη αλληλεπίδραση που έχουν μεταξύ τους οφείλεται κυρίως σε περιοδικά μηνύματα που στέλνονται από το Akka, Gossip και Raft πρωτόκολλο, όπως heartbeat μηνύματα.

Πειραματισμοί πάνω στο Inventory Shard

Σε αυτή τη σειρά των πειραμάτων, μελετήθηκε η συμπεριφορά του συμπλέγματος όταν εισάγεται νέα πληροφορία στο inventory shard. Το inventory shard περιέχει τα flows που εγκαθίστανται στο σύμπλεγμα. Ο αριθμός των switches παραμένει στα τρία, το καθένα εκ των οποίων είναι συνδεδεμένο με έναν controller. Η Εικόνα 5 απεικονίζει τις τοπολογίες που ρυθμίστηκαν κατα την διάρκεια αυτών των πειραμάτων. Τα flows εγκαταστάθηκαν με την εντολή ovs-vsctl, η οποία χρησιμοποιείται γενικά για διαχείριση και παρακολούθηση των OpenFlow switches.

Αρχικά, αυξήθηκε ο αριθμός των flows που είναι εγκατεστημένα στο switch που είναι συνδεδεμένο με τον leader κόμβο. Παρόμοια με την περίπτωση του Topology Shard, όταν εισάγονται νέα δεδομένα στο leader inventory shard, παρατηρείται αύξηση στην χρήση bandwidth από την πλευρά του leader προς τους follower κόμβους, ενώ η επικοινωνία από την πλευρά των followers παραμένει σταθερή. Το bandwidth που μετρήθηκε σε αυτό το πείραμα παρουσιάζεται στον Πίνακα 2, όπου η πρώτη στήλη πλέον ονομάζεται #flows και δείχνει τον αριθμό των flows στο switch του κάθε controller. Οι στήλες avg_1 και avg_2 απεικονίζουν την μέση αύξηση στο bandwidth για κάθε flow που εισάγεται στο switch που ελέγχεται από τον f_1 και f_2 αντίστοιχα. Τα avg_1 και avg_2 συγκλίνουν στο $0,002Mbps$, όπως μπορούμε να δούμε στις τελικές τιμές του πρώτου μισού του Πίνακα 2.



Εικόνα 5 - #flows flows ριθμίζονται στο μοναδικό switch που βρίσκεται συνδεδεμένο είτε στον Leader (πρώτη φάση), είτε στον Follower 1 (δεύτερη φάση), για την μέτρηση της Ctr-Sw και Ctr-Ctr κίνησης που δημιουργείται από το Inventory Shard.

#flows		bandwidth usage (Mbps)							
l	f_1	$l \rightarrow f_1$	$f_1 \rightarrow l$	$l \leftrightarrow f_1$	avg_1	$l \rightarrow f_2$	$f_2 \rightarrow l$	$l \leftrightarrow f_2$	avg_2
0	0	1.00	1.10	2.10	–	1.00	1.10	2.10	–
30	0	1.04	1.10	2.14	0.001	1.02	1.10	2.12	0.001
60	0	1.08	1.10	2.18	0.001	1.08	1.10	2.18	0.001
120	0	1.19	1.10	2.29	0.002	1.19	1.10	2.29	0.002
240	0	1.45	1.10	2.55	0.002	1.45	1.10	2.55	0.002
0	30	1.09	1.25	2.34	0.003	1.09	1.10	2.19	0.003
0	60	1.21	1.35	2.56	0.004	1.09	1.10	2.19	0.002
0	120	1.46	1.60	3.06	0.004	1.20	1.10	2.30	0.002
0	240	1.85	2.03	3.88	0.004	1.39	1.10	2.49	0.002

Πίνακας 2: Αποτελέσματα πειραμάτων στο Inventory Shard
Χρήση bandwidth για διάφορους αριθμούς flows σε κάθε switch.

Στη δεύτερη φάση, flows εγκαθίστονται στο switch που είναι συνδεδεμένο στον Follower 1. Για μία ακόμα φορά, είναι εμφανής ο κυρίαρχος ρόλος του Leader κόμβου στην αντιγραφή των καινούργιων δεδομένων προς τους υπόλοιπους κόμβους. Ο Πίνακας 2 δείχνει ότι υπάρχει μία αύξηση στο bandwidth από τον Follower 1 προς τον Leader, ενώ ο Follower 1 προωθεί τα νέα δεδομένα στον Leader. Ο Leader κόμβος αμέσως προωθεί τα δεδομένα στους υπόλοιπους followers, το οποίο έχει ως αποτέλεσμα την αύξηση στο bandwidth που παρατηρείται στις στήλες $l \rightarrow f_1$ και $l \rightarrow f_2$. Οι τελικές τιμές των avg_1 και avg_2 στηλών συγκλίνουν στα $0,004Mbps$ και $0,002Mbps$ αντίστοιχα. Άρα, επαληθεύεται και η **Παρατήρηση 2**, έχοντας ως αποτέλεσμα τις τιμές $\beta_{inv}^{int} \approx 0,004\text{M}bps$ και $\beta_{inv}^{ext} \approx 0,002\text{M}bps$.

Επικοινωνία μεταξύ Ελεγκτή και Switch

Σε αυτή τη σειρά πειραμάτων, εξετάζεται η Ctr-Sw κίνηση. Τα αποτελέσματα των πειραμάτων βρίσκονται στον Πίνακα 3, όπου οι στήλες $ctr \rightarrow sw$ και $sw \rightarrow ctr$ αναφέρονται στην κίνηση που εισάγεται από τον controller στο switch και το ανάποδο. Σε αυτό το σετ πειραμάτων, η τοπολογία παραμένει ίδια όπως αυτή της πρώτης φάσης που απεικονίζεται στην Εικόνα 5, αυξάνοντας τον αριθμό των flows που είναι εγκατεστημένα στο switch που είναι

συνδεδεμένο στον Leader και παρακολουθείται η $ctr \rightarrow sw$ και $sw \rightarrow ctr$ κίνηση. Η μέση αύξηση για κάθε ένα flow που εισάγεται, όπως φαίνεται στην avg στήλη, είναι 0,18Kbps. Έτσι, επιβεβαιώνεται η Παρατήρηση 1 και $\beta^s \approx 0.18\text{K}bps$. Σε αυτό το σημείο αξίζει να σημειωθεί πως το β^s είναιη κίνηση που παράγεται από την εισαγωγή ενός flow εξαιτίας των στατιστικών που αποθηκεύονται από τον controller για αυτή την εισαγωγή. Αυτή είναι η μακροχρόνια μέση κίνηση ελέγχου που παράγεται από κάθε ξεχωριστή εισαγωγή flow, η οποία μπορεί να είναι πολύ μικρότερη από την κίνηση που παράγεται την στιγμή την οποία το flow εγκαθίσταται στο switch. Επιπλέον, τα switch μπορούν να συνδεθούν σε πολλαπλούς controllers, αυξάνοντας την ανθεκτικότητά τους απέναντι σε σφάλματα των ελεγκτών. Ο controller που προτείνεται από την λύση του προβληματος μας είναι ο master, ενώ οι υπόλοιποι controllers αποκαλούνται slaves. Τα αποτελέσματα των πειραμάτων δείχνουν πως η κίνηση ελέγχου μεταξύ των switch και των slaves ελεγκτών είναι πολύ μικρότερη και αμελητέα, συγκρίνοντάς τη με την υπόλοιπη κίνηση ελέγχου.

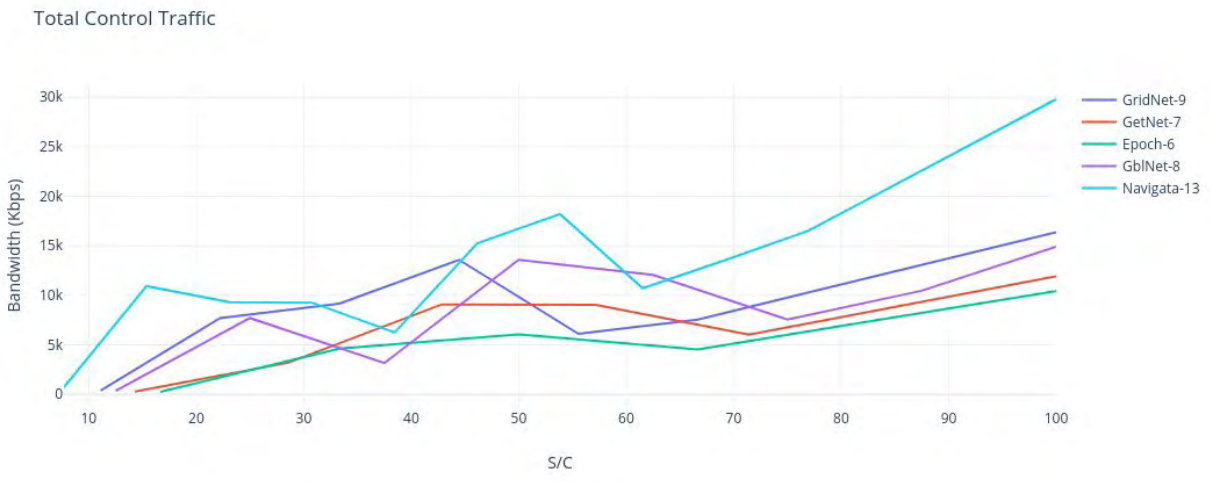
#flows	*bandwidth usage (Kbps)*			
	sw → ctr	*ctr → sw*	*sw ↔ ctr*	*avg*
0	18.10	1.88	19.98	–
10	19.80	1.92	21.72	0.17
20	21.53	1.95	23.48	0.18
50	26.73	2.04	28.77	0.18
100	36.00	2.19	38.19	0.18
200	53.90	2.50	56.40	0.18

Πίνακας 3: Ctr-Sw αποτελέσματα πειραμάτων. Χρήση bandwidth για διάφορους αριθμούς flows σε κάθε switch.

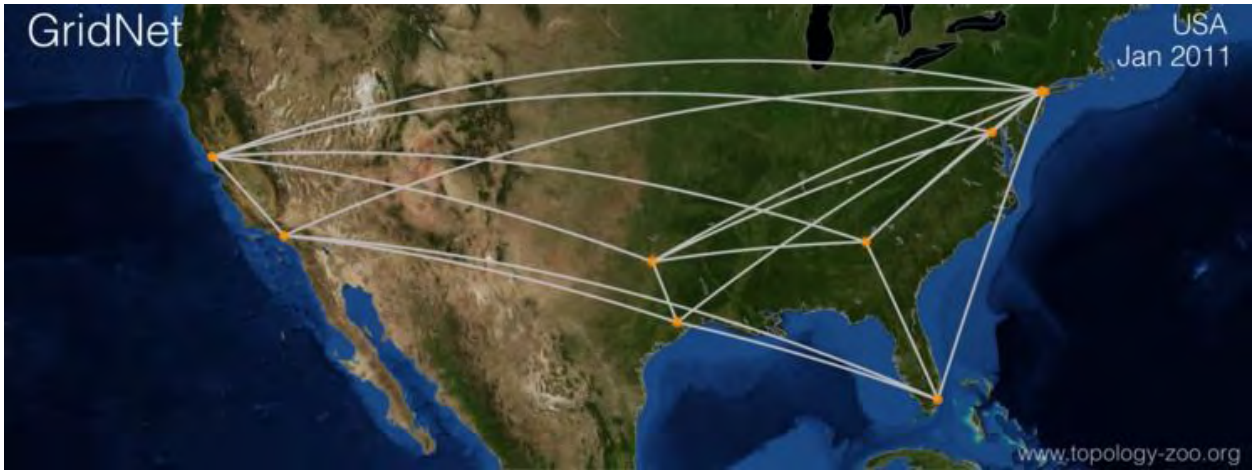
Κίνηση ελέγχου και Ανθεκτικότητα Δικτύου

Εδώ χρησιμοποιείται το μοντέλο δικτύου που περιγράψαμε σε προηγούμενη ενότητα, με σκοπό να προσομοιωθεί η Ctr-Ctr και Ctr-Sw κίνηση σε διάφορες τοπολογίες δικτύου. Το μοντέλο χτίστηκε βασισμένο στις τιμές για τα β_{topo}^{ext}, β_{topo}^{int}, β_{inv}^{ext}, β_{inv}^{int} και β^{s} οι οποίες προέκυψαν από τα παραπάνω πειράματα. Το IQP πρόβλημα της Εξίσωσης 5 λύθηκε χρησιμοποιώντας την γλώσσα προγραμματισμού R [14] και η βιβλιοθήκη RCPLEX [15] σε διάφορες τοπολογίες δικτύου, τις οποίες παρέχει το Internet Topology Zoo Collection [16].

Εξετάζοντας τα αποτελέσματα αυτών των πειραμάτων, παρατηρήθηκε πως η ελάχιστη κίνηση ελέγχου προκύπτει όταν υπάρχει ένας μοναδικός ελεγκτής στο δίκτυο. Ωστόσο, ένα δίκτυο που λειτουργεί μόνο με έναν ελεγκτή έχει μηδενική ανθεκτικότητα σε σφάλματα του ελεγκτή. Ένα ελάχιστο των τριών ελεγκτών απαιτείται ώστε να δημιουργηθεί ένα σύμπλεγμα ανθεκτικό σε τέτοια σφάλματα. Η Εικόνα 6 δείχνει τη συνολική κίνηση ελέγχου για διάφορα %C/S σε δίκτυα με τυχαίους συνδέσμους μεταξύ των κόμβων, ενώ η Εικόνα 7 δείχνει μια από τις τοπολογίες τέτοιου δικτύου με τυχαίους συνδέσμους όπου συμμετείχαν στα πειράματα. Καθώς αυξάνεται η ανθεκτικότητα του δικτύου, η κίνηση ελέγχου δεν ακολουθεί γραμμική αύξηση. Οπότε, ένα νέο ελάχιστο μπορεί να βρεθεί για την κίνηση ελέγχου σε κάθε δίκτυο, για αριθμό ελεγκτών μεγαλύτερο από τρία.

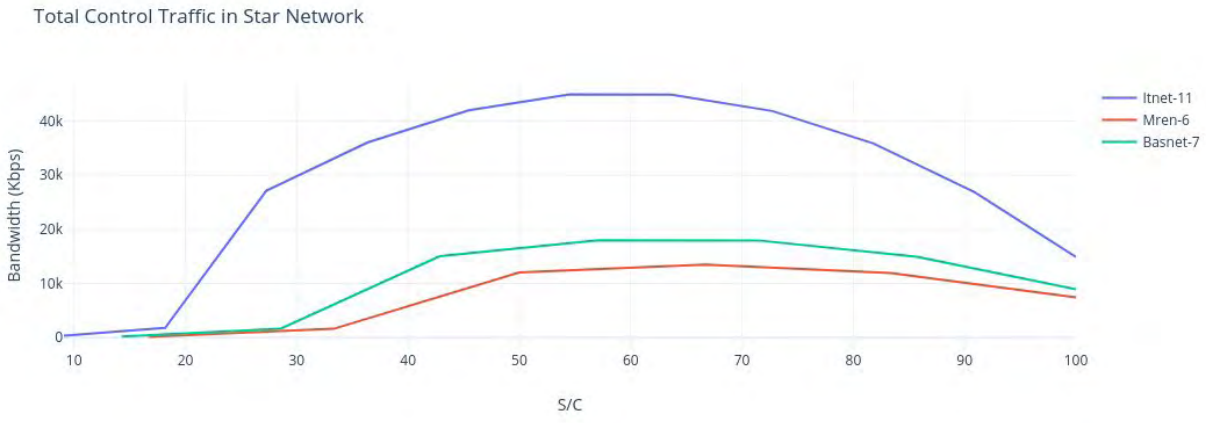


Εικόνα 6: Στα δικτυα με τυχαίους συνδέσμους μεταξυ των κομβων, σε ένα ποσοστό φτανει στο ελάχιστο και απο κει και περα αυξάνεται η κίνηση ελέγχου

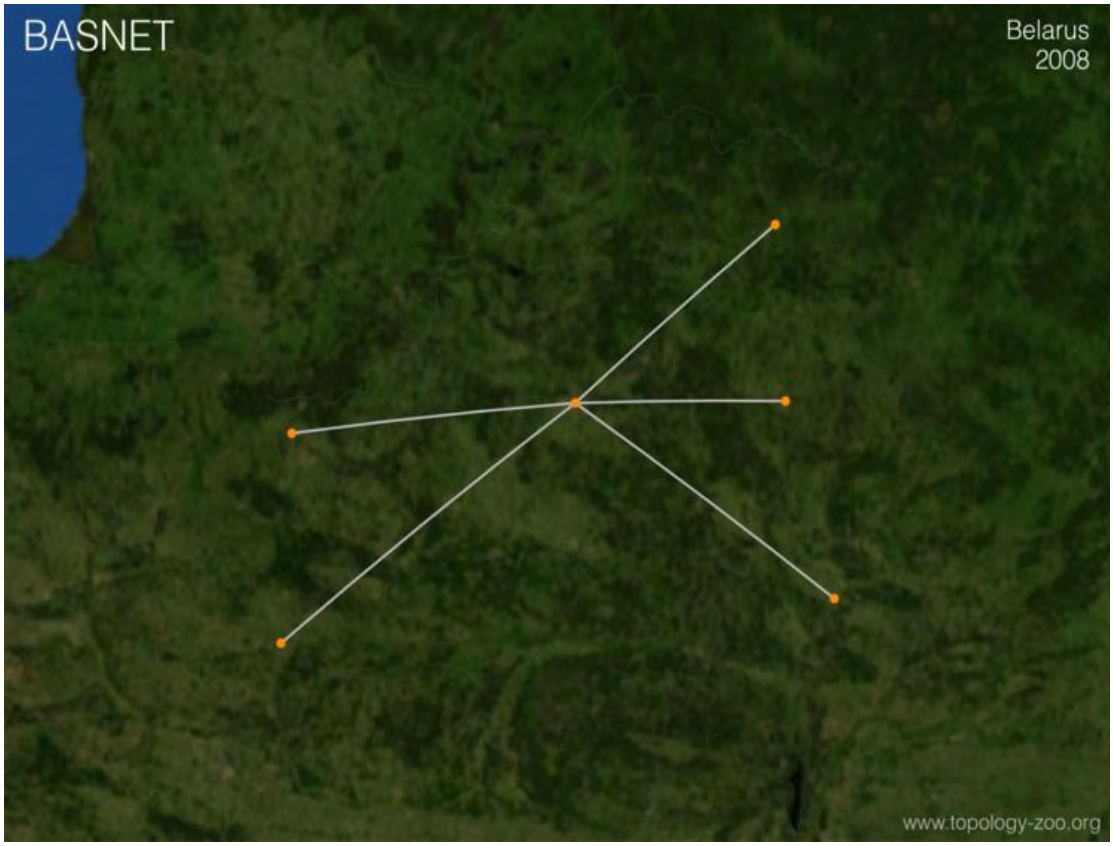


Εικόνα 7: Απεικόνιση τοπολογίας δικτύου με τυχαίους συνδέσμους μεταξύ των κομβων, όπου συμμετείχε στα παραπάνω πειράματα

Η Εικόνα 8 δείχνει τη συνολική κίνηση ελέγχου για διάφορα %C/S σε δίκτυα Star τοπολογίας, ενώ η Εικόνα 7 δείχνει μια από τις τοπολογίες τέτοιου δικτύου όπου συμμετείχαν στα πειράματα.
Στα Star δικτυα, η κίνηση ελέγχου ελαχιστοποιείται όταν τοποθετούνται controllers σε ολα τα switches, δηλαδη #ctr = #sw.

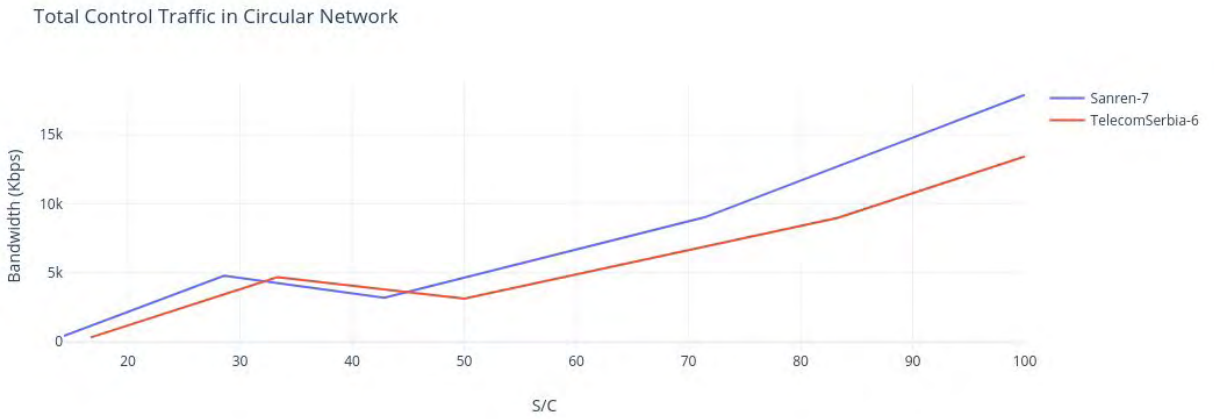


Εικόνα 8: Στα Star δικτυα, η κίνηση ελέγχου ελαχιστοποιείται όταν ελεγκτές τοποθετούνται σε όλα τα switches του δικτύου.



Εικόνα 9: Απεικόνιση Star τοπολογίας δικτύου, όπου συμμετείχε στα παραπάνω πειράματα

Η Εικόνα 10 δείχνει τη συνολική κίνηση ελέγχου για διάφορα %C/S σε δίκτυα κυκλικής τοπολογίας, ενώ η Εικόνα 11 δείχνει μια από τις τοπολογίες τέτοιου δικτύου όπου συμμετείχαν στα πειράματα.
Από την Εικόνα 10 συμπεραίνεται πως στα κυκλικά δικτυα, η κίνηση ελέγχου ελαχιστοποιείται όταν τοποθετούνται controllers σε ακριβώς τρία switches, δηλαδη #ctr = 3.



Εικόνα 10: Στα κυκλικά δικτυα, η κίνηση ελέγχου ελαχιστοποιείται όταν ελεγκτές τοποθετούνται σε ακριβώς τρία switches του δικτύου.

Εικόνα 11: Απεικόνιση κυκλικής τοπολογίας δικτύου, όπου συμμετείχε στα παραπάνω πειράματα

Επίλογος

Σε αυτή την έρευνα, μελετήθηκαν οι επιδράσεις λειτουργίας ενός συμπλέγματος ODL ελεγκτών στο δίκτυο, αντί για έναν μοναδικό ελεγκτή. Αρχικά, έγινε μία ανασκόπηση στην λογική πίσω από την κατανομή των δεδομένων μεταξύ των κόμβων του του συμπλέγματος, ονομαζόμενη Distributed Datastore. Έπειτα, αναφέρεται ο ρόλος του μηχανισμού Akka, ακολουθούμενος από τη περιγραφή του πρωτοκόλλου Gossip και του αλγορίθμου Raft. Στην συνέχεια, δημιουργείται ένα μοντέλο δικτύου το οποίο περιγράφει τη συνολική κίνηση ελέγχου που δημιουργείται μέσα σε ένα σύμπλεγμα ODL ελεγκτών, βασιζόμενο στα αποτελέσματα των πειραμάτων. Κατα την διάρκεια των πειραμάτων, μετρήθηκε η επιβάρυνση που εισήχθει στη χρήση bandwidth του δικτύου εξαιτίας της επικοινωνίας των ελεγκτών που γίνεται μεταξύ τους. Παρακολουθήθηκαν οι αλλαγές στη χρήση bandwidth μέσα στο σύμπλεγμα ως αποτέλεσμα της εγκατάστασης νέων flows και της σύνδεσης νέων switches στους ελεγκτές.

Καταλήγοντας σε ένα συμπέρασμα, το μαθηματικό μοντέλο όχι μόνο περιγράφει την χρήση bandwidth μέσα στο ODL σύμπλεγμα λεπτομερώς, αλλά επίσης τονίζει τον σημαντικό ρόλο που έχει ο leader κόμβος ως την καρδία του συμπλέγματος. Το μοντέλο δικτύου μπορεί να φανεί χρήσιμο στην λήψη αποφάσεων για το πως θα τοποθετηθούν οι ελεγκτές του συμπλέγματος στο δίκτυο, βασιζόμενοι στο ρόλο του καθενός.

Βιβλιογραφία

[1] N. Feamster, J. Rexford and E. Zegura, “The road to SDN”, ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 87– 98, 2014.

[2] J. Medved, R. Varga, A. Tkacik and K. Gray, “OpenDaylight: Towards a Model-Driven SDN Controller architecture”, Proc. of IEEE WoWMoM 2014, Sydney, NSW, Australia, June 2014.

[3] K. Choumas, D. Giatsios, P. Flegkas and T. Korakis, “The SDN Control Plane Challenge for Minimum Control traffic: Distributed or Centralized?”, Proc. of IEEE CCNC 2019, Las Vegas, USA, January 2019.

[4] J. Esch, “Prolog to, ‘software-defined networking: a comprehensive survey,’” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10–13, 2015.

[5] L. Foundation, “Opendaylight Platform Overview,” 2016. [Online]. Available: https://www.opendaylight.org/platform-overview

[6] L. Foundation, “Opendaylight Platform Overview,” 2016. [Online]. Available: https://www.opendaylight.org/

[7] M. Raja, “Md sal clustering internals,” *LinkedIn SlideShare*, 29-Jul-2015. [Online]. Available: https://www.slideshare.net/moizhraja/md-sal-clustering-internals.

[8] D. Ongaro and J. Ousterhout, “In Search of an Understandable Consensus Algorithm”, ATC 14, vol. 22, no. 2, pp. 305–320, 2014

[9] Akka Cluster: https://doc.akka.io/docs/akka/2.5/ common/cluster.html

[10] T. Kim, S.-G. Choi, J. Myung and C.-G. Lim, “Load balancing on distributed datastore in opendaylight SDN controller cluster”, Proc. of IEEE NetSoft 2017, Bologna, Italy, July 2017

[11] NITOS: https://nitlab.inf.uth.gr/NITlab/nitos

[12] Mininet: http://mininet.org/

[13] Iftop: https://en.wikipedia.org/wiki/Iftop

[14] The R project for Statistical Computing. http://www.r-project.org

[15] IBM ILOG CPLEX for Faculty. http://www01.ibm.com/software/commerce/optimization/cplex-optimizer.

[16] The Internet Topology Zoo. http://www.topology-zoo.org/dataset.html.